MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1323 October 1991

A Control Algorithm for Chaotic Physical Systems

Elizabeth Bradley

Abstract

Control algorithms that exploit chaos's unique properties can vastly improve the performance of many practical and useful systems. The program *Perfect Moment* is built around such an algorithm. Given a differential equation and two points in the system's state space, it automatically maps the space, chooses a set of trajectory segments from the maps, uses them to construct a composite path between the points, and causes the system to follow that path by monitoring the state and switching parameter values at the segment junctions. The creation of and search through the maps are computationally intensive processes. However, the sensitivity of a chaotic system's state-space topology to the parameters of its equations and the sensitivity of the paths of its trajectories to the initial conditions make this approach rewarding in spite of its computational demands. This program and its results are illustrated with several examples, among them the driven single pendulum and its electronic analog, the phase-locked loop. In this particular case, strange attractor bridges, which traverse boundaries of basins of attraction and thus alter the reachability of different state space points, can be used to broaden the capture range of the circuit.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-89-J-3202 and by the National Science Foundation under grant number MIP-9001651. The author is supported by an American Association of University Women dissertation fellowship.

This paper was presented in October 1991 at the First Experimental Chaos Conference in Washington D.C. Proceedings: World Scientific.

A Control Algorithm for Chaotic Physical Systems

Elizabeth Bradley
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

2 October 1991

Abstract

Control algorithms that exploit chaos's unique properties can vastly improve the performance of many practical and useful systems. The program *Perfect Moment* is built around such an algorithm. Given a differential equation and two points in the system's state space, it automatically maps the space, chooses a set of trajectory segments from the maps, uses them to construct a composite path between the points, and causes the system to follow that path by monitoring the state and switching parameter values at the segment junctions. The creation of and search through the maps are computationally intensive processes. However, the sensitivity of a chaotic system's state-space topology to the parameters of its equations and the sensitivity of the paths of its trajectories to the initial conditions make this approach rewarding in spite of its computational demands. This program and its results are illustrated with several examples, among them the driven single pendulum and its electronic analog, the phase-locked loop. In this particular case, strange attractor bridges, which traverse boundaries of basins of attraction and thus alter the reachability of different state space points, can be used to broaden the capture range of the circuit.

1 Introduction

Control algorithms that exploit the special properties of chaotic systems open a new category of problems to effective design — interesting and useful applications whose performance is rich but whose analysis is mathematically and computationally demanding. Traditional control theory simply avoids chaotic zones, which often comprise a large and rich part of a chaotic system's state space. The aim of this work is to actively *use* chaotic behavior to improve performance. The algorithms presented in this paper intentionally route systems through chaotic zones, using extensive simulation, qualitative and quantitative reasoning about state-space features and heuristics drawn from nonlinear dynamics theory to navigate through the state space. Trajectory segments are selected from a collection of automatically-constructed state-space maps and pieced together into a path between the specified states. The controller causes the system to follow these segmented paths by monitoring the system state and switching parameter values when the segment junctions are reached.

These algorithms can be applied to any system, but are designed to exploit several specific properties of nonlinearity and chaos. For example, the denseness with which chaotic trajectories cover a section of state space — the strange attractor — has obvious implications for reachability. The classic "sensitive dependence on initial conditions" endows small control actions with large, global effects. Topological changes induced in a system's state space portrait by parameter variations add another degree of leverage to the picture; small changes in the control parameter can give the path-finding algorithm a large range of behaviors to exploit.

Striking results have been achieved with these algorithms[2]: a very small control action, delivered precisely at the right time and place, can accurately direct the system to a distant point on the state space — hence the name *Perfect Moment.* An equally small change can be used to move a particular state from the basin of attraction of one fixed point to the basin of another. Control actions can briefly push a system directly away from the goal state in order to reach a globally superior path to that state and "strange attractor bridges" can open conduits between previously-unreachable regions.

2 Context

The equations for an n-dimensional nonlinear system can be written

$$\frac{d\vec{x}}{dt} = \vec{F}(\vec{x}, k_1, \ldots, k_p, t) \tag{1}$$

where $\vec{x}$ is an n-vector of system state variables and $\vec{F}$ is a nonlinear function of the state $\vec{x}$ and the parameters k_i. Necessary conditions for chaos — in continuous-time systems – are that $\vec{F}$ be nonlinear and nonintegrable and that the dimension of the state space be at least three. The distance between state-space trajectories of a dissipative chaotic system grows exponentially with time, yet the trajectories remain on a bounded subset of the state space, called a strange attractor, within which are embedded an infinite number of unstable periodic orbits. The largest positive Lyapunov exponent λ of the system governs the growth of the distance between trajectories. The parameters k_i can strongly affect the direction and magnitude of the vector field $\vec{F}$; small changes in these parameters can have local and global effects on state-space features, causing fixed points to multiply or to mutate into other kinds of attractors. Between these topological changes or *bifurcations*, parameter changes can also cause dramatic changes in the position, shape and size of existing attractors. More details about the concepts in this paragraph may be found in [5].

These properties make possible a new spectrum of nonlinear control design techniques which derive their power from *not* treating the system as a linearized cousin to its true nature, as do most traditional techniques. This field has only recently seen development; the first generation of schemes therein exploit the denseness of strange attractors and of unstable periodic orbits[4, 7], but many other tacks are also possible. State-space regions where the system is sensitively dependent on initial conditions, once identified and characterized, can be used to magnify the effects of small control actions. The statistical properties of strange attractors determine the robustness of control schemes that use them. To make a point reachable, one need only identify parameter values that cause a strange attractor covering that point to appear; the parameter sensitivity of these features reduces the size of the control action necessary to accomplish this. This type of analysis and design requires extensive and detailed knowledge about the state-space features of the system under control. The first part of the next section discusses a computer algorithm that constructs a set of maps that detail these features and the second part describes a program that searches through them to find an ensemble of segments that together form a useful system trajectory.

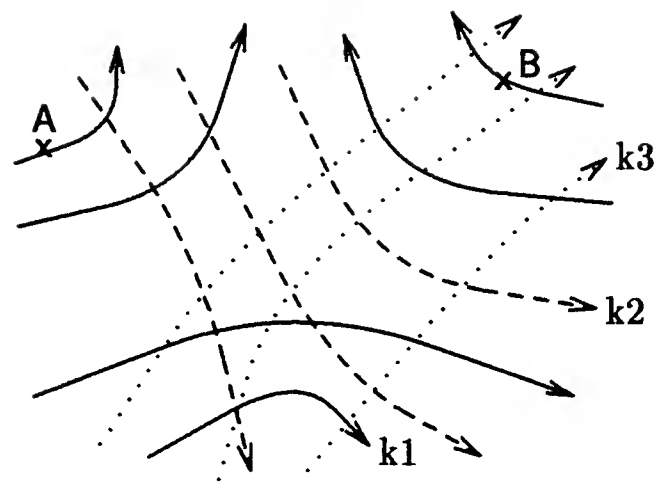

Figure 1: Finding a Path Through a Collection of State-Space Portraits: A = origin, B = destination.

3 Issues and Algorithms

Envision a stack of state-space maps, each plotted at a different value of the control parameter in a different color on a transparent sheet. Looking down through such a stack, one could easily choose a set of segments that together form a path from one point to another. See figure 1.

This task is relatively easy for a human expert and quite difficult for a computer. The ease with which human vision performs pattern recognition and processes coincident information on different scales, coupled with domain knowledge that allows interpretation, inference and prediction of the effects of parameter and state changes, makes state-space exploration intuitive and straightforward to a nonlinear dynamicist. One simply selects a "representative" set of trajectories on each of a "representative" set of maps, then looks through them for a "good" path. The quotes highlight the concepts that are difficult to mechanize. This section discusses solutions to these three problems; the algorithms that embody these solutions use the cell-to-cell mapping formalism of Hsu[6], which is reviewed briefly in the next paragraph. They are written in the Lisp dialect Scheme[8] and run on HP series-300 and series-700 workstations and on the Supercomputer Toolkit[1].

The state space is partitioned on a grid. Each of the n state variables $x_i, i = 1 \ldots n$ is divided into m_i equal intervals h_i, so the space contains $M = \prod_i m_i$ cells. Each parallelepiped in this grid is identified by an n-vector of coordinates z_i, called a cell vector, where the ith coordinate gives the grid height in the ith dimension. A state-space trajectory can be represented as a list of the cells touched, together with time of entry to and exit from each

cell; a description of the entire flow can be given as a set of M mappings, each of which describes the travels of a trajectory that starts at the center of a cell. Coarse discretization — large h_i — lowers accuracy, speeds searches and is useful in early stages of work.

Trajectory Selection and Generation

The first stage in the mapping process is the selection of a "representative" set of trajectories: enough to characterize the system's dynamics across the useful region of the state space, but no more than necessary to do so, as the speed of the later stages of the program degrades rapidly with the size of the search space.

A trajectory emanating from a point is generated by numerical integration of the system equations from that initial condition. *Perfect Moment* uses an adaptive-time-step fourth-order Runge-Kutta algorithm.

Selecting a representative set of trajectories from the uncountably infinite number of possible ones is much more subtle. One solution is to start a trajectory from the center of each grid square and integrate until it relaxes to an attractor or exits the grid. This requires a good choice of h_i: small enough so that each region is explored and yet large enough to restrict the amount of information to the minimum necessary to capture the characteristics of the space. The initial choice h_i^0 is currently made by the user and heuristically revised downwards by the program over the course of the mapping and path-finding process. This will eventually be fully automated, based on nonlinear dynamics knowledge. For example, only one trajectory in the basin of attraction of an asymptotically-stable fixed point need be plotted. Some related issues that arise in problems of this nature are addressed in much more detail in [9], wherein domain knowledge about nonlinear systems is used to infer when important trajectories are missing from state-space portraits.

The selection of the dimensions of the region under consideration is also subtle. If *Perfect Moment* restricted its attention to trajectories within the bounding box of the origin and destination points, counterintuitive moves — ones that send the system away from the apparently "correct" direction in order to reach a faster path — would be impossible. The heuristic overrange parameter R^0 allows for this; an R^0 value of 1.1 implies that the search region is formed by expanding the bounding box defined by the origin and destination points by 10% in all directions. Like h_i^0, this parameter is initially chosen by the user and revised dynamically by the program. One of the heuristics used in this revision expands the grid to cover the entire area of any attractor that is present.

Map Selection

The final choices of the state-space mapping process concern the control parameter: its interval between successive maps Δk and the total range over which the space is explored $[k_{low}, k_{high}]$. Like h_i, these numbers are user-chosen and automatically-revised: if successive maps at the specified parameter interval are significantly different — for example, if bifurcations have occurred or attractors have expanded or shrunk to cover or uncover the control objective or some other useful point — that interval is explored further. This magnification is iterated on a finer and finer scale until successive maps are similar or until a user-specified iteration depth is reached.

Segment Selection

After the mapping process is complete, the path-finder searches through the collection of maps for useful path segments, first on a coarse grain and then with iteratively-refined accuracy. Useful segments may be drawn from any region on any map — they may be sections of strange attractors or of transient trajectories that are relaxing to attractors. Optimization criteria are specified by the user and might include time, path length along one or more of the state variable axes, etc. Total time is minimized in all examples in this paper; changes to and generalizations of this choice are obvious and easy.

A gross trajectory between the *endpoint cells* of size h_L that contain the origin and the destination is found first and used as the core of the path. The parameter h_L is chosen according to the amount of "spreading" undergone by trajectories emanating from the origin. It is initially determined by rules that use Lyapunov exponents and integrations of the variational system from the origin and is revised upwards if the search fails. The cell size is then reduced and the process is repeated within the endpoint cells to connect the core segment to the origin and destination. This strategy allows for counterintuitive moves within both endpoint cells *at any level of the search.* It also reduces the work: the finer discretization makes the later searches slower, but they occur in smaller regions. The critical steps here are (1) the reduction of h_i between iterations and (2) the termination condition. If h_L and R^0 are such that the grid contains c_i cells on a side, the next-level search divides the endpoint cell into $\prod_i c_i$ cells, each with the same aspect ratio as the endpoint cell. The path-finding process halts when the error introduced by omitting the smallest segment, integrated over the entire path, falls within the control tolerance. If the length of the segment first falls below the machine epsilon or below the effective resolution of any physical I/O devices, the program is restarted with different heuristics.

Effecting Control

The system state can be caused to evolve along a trajectory consisting of a series of path segments $\{S^0, \ldots, S^l\}$ via the following set of control actions. Because of the recursive, longest-first nature of the path-finding algorithm, the segments are not followed in the order in which they are found, so the list must first be sorted into the proper order $\{S'^0, \ldots, S'^l\}$. Beginning at the specified origin, the control parameter is set to k'_0 to initiate the first segment S'^0 and the state is monitored until $\vec{x} = S'^0_{final}$. The parameter is then changed to k'_1, rerouting the system onto S'^1. This procedure is repeated through all segments in the path. After the final switch, if necessary, the system can be stabilized at the destination by a linearized control scheme[2, 7].

It is vital that parameter switches take place quickly and accurately, both in time and in magnitude. Speed is a formidable limit here, since control actions moderated by computer I/O must occur roughly ten times faster than the natural frequencies of the system under control. Mechanical devices are well within *Perfect Moment*'s current grasp and control of circuits with bandwidths on the order of kilohertz is conceivable with current technology[1]. Quantization, experimental and modeling error are all but unavoidable; the optical encoder currently used to measure the position of the pendulum described in section 4, for example, has an angular resolution of 0.7 degree. Since the ultimate goal of this project is to control real physical systems, it uses a programmable "backdoor" correction network to compensate for such errors — a simple autonomous linear controller programmed with the linearization at each segment junction point.

Problems and Caveats

A variety of conditions can cause these algorithms to fail. Most importantly, the system state must be observable, either directly or indirectly. Though the control parameters add dimensions to the space that can open conduits between previously-unreachable regions, some destinations may still be unreachable — repellors which persist for all parameter values, for instance. Small control tolerances are fundamentally unachievable in the face of huge Lyapunov exponents, inaccurate computer arithmetic, slow I/O or bad models.

Perfect Moment currently handles systems that have a single control parameter. This could easily be generalized, but a higher-dimensional parameter space would slow the mapping and search processes and introduce no new ideas or techniques, so this angle is not pursued here.

This program relies on pre-simulation. None of the algorithms detailed in this section run in real time, nor could they: many hours of CPU time are

required for their integrations and searches. *Perfect Moment*'s intended use is to find paths that will be followed many times or paths that are particularly important.

Synopsis

- Inputs: system equation; origin **A** and destination **B**; optimization criteria; control tolerance; grid interval h_i^0; overrange R^0; control parameter interval Δk^0 and range $[k_{low}, k_{high}]$; iteration depth.
- State-space mapping:

  1. Set up grid by expanding bounding box of **A** and **B** by the factor R^0. Construct state-space portraits at each $k_{low} + n\Delta k$ for $(k_{low} + n\Delta k) \in [k_{low}, k_{high}]$, starting a trajectory at the center of each cell.
  2. Revise R if required by attractor size.
  3. Construct portraits with half the Δk in ranges where successive plots exhibit large differences. Iterate this step up to the specified iteration depth.

  This mapping will only be repeated if the path-finder below fails to find an adequate solution.

- Path-Finding:

  1. Using these maps, variational integrations from **A** and the grid interval h_i^0, find the shortest path between the endpoint cells of size h_L containing **A** and **B**. This segment is designated S^0; it starts at S^0_{init} and ends at S^0_{final} with $k = k_0$.
  2. Reduce h_i and find the best path between the cells containing (1) S^0_{init} and **A** and (2) **B** and S^0_{final}. Iterate on successively smaller scales until the the last segment causes the path to exceed the control tolerance. Record the k_i value and the starting and ending states S^i_{init} and S^i_{final} of each segment.
  3. If the search fails, repeat with heuristically modified h_i and inter-step reduction factor.
  4. If step 3 fails repeatedly, redo the mapping with a larger R^0, wider parameter range $[k_{low}, k_{high}]$ and greater iteration depth and restart the path-finding program.
  5. Sort the segments into the proper order and generate the control recipe: the ordered list of parameter values and endpoints for each.

4 Examples

All of the examples in this section are simulated, but the models and controls reflect the physical parameters of a mechanical pendulum and a phase-locked

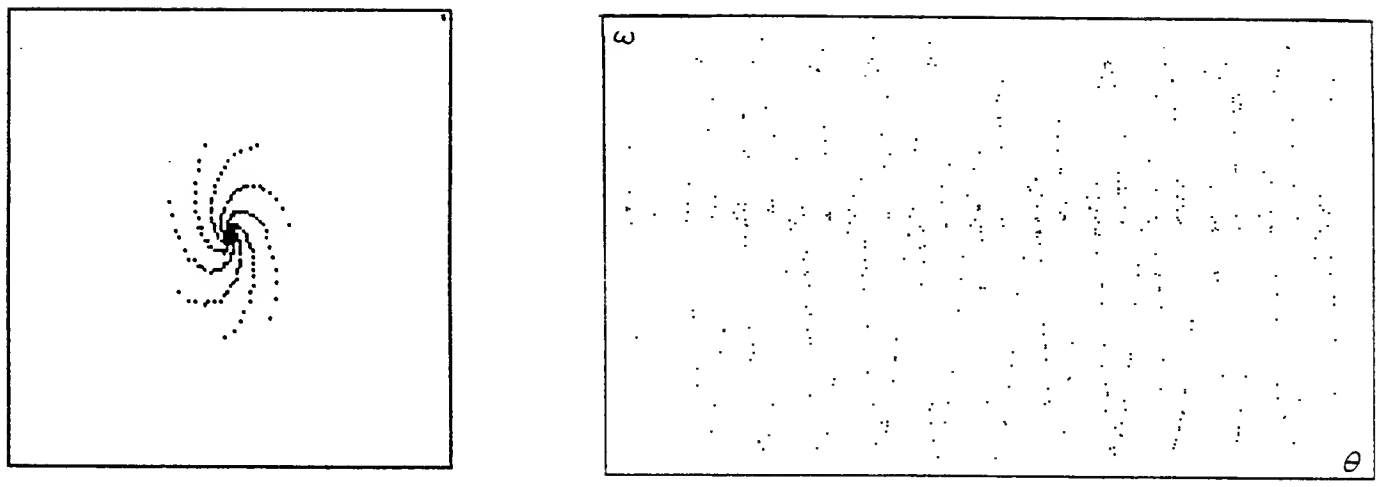

Figure 2: Driven Pendulum Poincaré Sections: (a) Damped Oscillations with Small Drive (b) Chaotic Behavior with Larger Drive

loop circuit that have been constructed as test cases for these techniques and are in the process of being instrumented.

The driven pendulum is arguably the most closely-studied simple chaotic system[3]. Its normalized equation is

$$\frac{d^2\theta}{dt^2} = \gamma(t) - \beta\frac{d\theta}{dt} - \sin\theta \qquad (2)$$

with angular position θ and velocity $\frac{d\theta}{dt}$, applied torque per unit mass $\gamma(t) = \epsilon gl\cos\omega_d t$ at the pivot and damping ratio (2 × actual/critical) β. ω_d is the ratio of the forcing frequency to the natural frequency and the normalized time unit is the inverse of the natural frequency. Both ϵ and ω_d play roles in chaotic state-space changes and thus can be used as parameters. In the experimental setup constructed for this project, however, changing ϵ is much more difficult, so ω_d is used as the control parameter and ϵ is fixed at 1.2. The damping β is also fixed at 0.007 throughout.

For small-amplitude drives or drive frequencies near the natural frequency, the pendulum remains "locked" to its natural frequency; the damping causes the oscillations to slowly die out, as shown in the Poincaré section of figure 2(a). Larger ϵ and smaller ω_d cause chaotic behavior, as in figure 2(b).

If the pendulum is started from the initial state **A** = (0,5) with no applied torque, the system follows the trajectory shown in figure 3(a). *Perfect Moment* was used to construct a segmented path to the control objective **B** = (-.5, -3.3) in this figure. The program was given a tolerance of 2%, cell size $h_i^0 = 0.05$; and a range [10, 20] and step $\Delta\omega_d = 1$ for the parameter ω_d. As this range encompasses no bifurcation regions, the mapper never

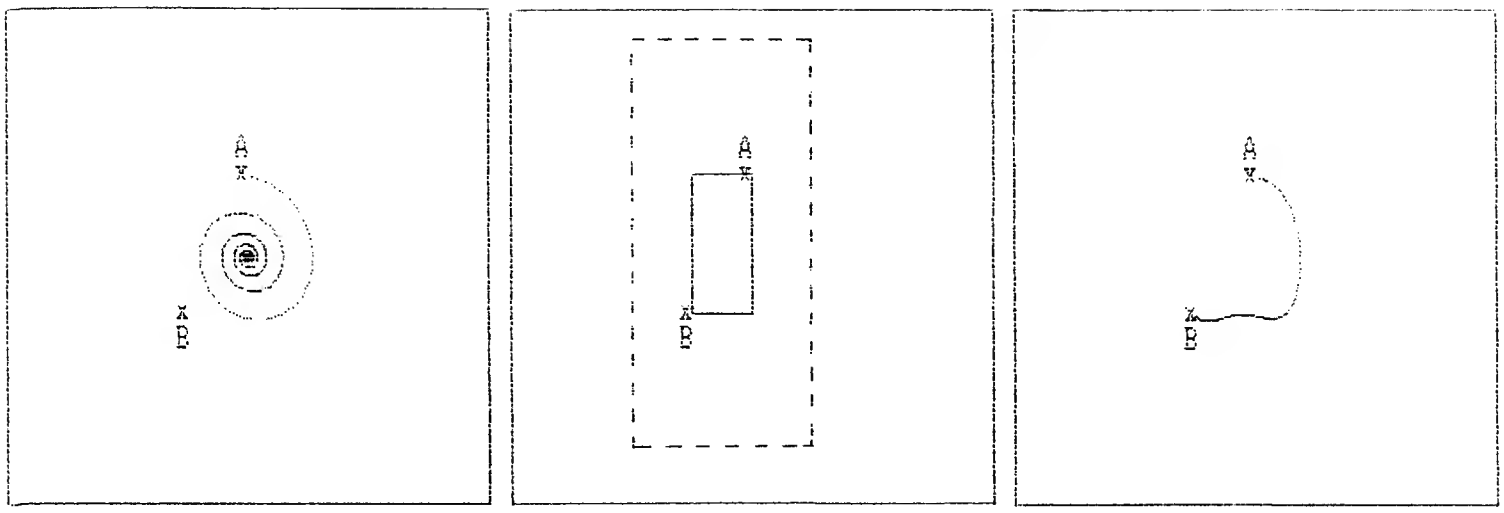

Figure 3: Driven Pendulum (a) Uncontrolled Trajectory (b) Search Regions (c) Controlled Trajectory

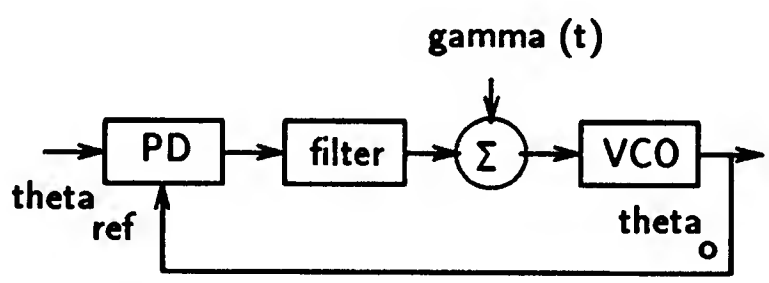

Figure 4: Phase-Locked Loop Block Diagram

reduced $\Delta\omega_d$ and so produced eleven equally-spaced maps of the region defined by expanding the bounding box of (0,5) and (-.5,-3.3) by $R = 3$; see the solid and dashed boxes on part (b) of the figure. The variational system integrated forwards from **A** indicated a choice of $h_L = 2h_i$ for the initial length of the ω_d-edge of the endpoint cells. The fastest trajectory segment between those two cells was found on the map constructed for $\omega_d = 15$. The cell dimensions were then reduced in such a way that each endpoint cell was divided on the same grid as the original region in the dashed box and the search was repeated. The two segments found in this search caused the path to meet the specified tolerance, so no further iterations were required. The total path, shown in figure 3(c), contains three segments but the two smaller ones are invisible because of the magnitude of the cell-size step between the first and second pass of the path-finding algorithm and the scale of the figure. Stabilization of the system at **B** is discussed elsewhere[2, 7].

Equation 2 also describes the dynamics of a particular kind of phase-locked loop (PLL) circuit. A block diagram of a PLL-based phase modulator is shown in figure 4. The difference between the output and reference phase is filtered and used to drive the voltage-controlled oscillator (VCO) so as to

minimize the difference $(\theta_o - \theta_{ref})$ and "lock" the output phase to that of the reference signal. The phase detector (PD), often a simple mixer, is the main source of the nonlinearity and of the harmonics that necessitate the filter. The voltage into the summing box, which corresponds to the applied torque in the previous example, modulates the output phase θ_o. Once a PLL is locked to a sinusoidal signal, changes in frequency can be tracked over some *lock range*. Initial lock can be acquired over a smaller *capture range*. These ranges depend in a complicated way on the global, nonlinear properties of the loop and define a rectangular region on the system's state-space. The forward path gain puts an upper bound on the former; the latter is smaller because of the filter's rolloff.

Perfect Moment can exploit the phase-modulating input $\gamma(t)$ to improve the capture range of the circuit. This usage is different in spirit from the paths described in the previous example, as the control objective is a wide range and not a single point. The best approach to this task is to use *Perfect Moment* to find a drive frequency ω_d^0 which, in conjunction with a particular reference frequency $\omega_{capture} \leq \omega_x \leq \omega_{lock}$, creates a strange attractor that overlaps the capture range limits of rectangle. Whenever an input reference frequency of ω_x is detected, an external drive with $\omega_d = \omega_d^0$ is applied until the system enters the rectangle. The drive is then immediately turned off, allowing the circuit to settle within the existing lock range. The strange attractor bridge found by the program effectively expands the capture range to include the entire lock range.

One final, brief example is presented here to demonstrate *Perfect Moment*'s use of counterintuitive moves. In a mathematically unrelated but familiar example, the Lorenz system, a strange attractor segment has been used to bridge boundaries of basins of attraction between two fixed points[2]. A trajectory emanating from the right-hand starting point in figure 5 — at a low parameter value — would normally converge to the left hand fixed point along the tightly-wound spiral at the bottom left of the figure. The right-hand path in the figure was found by a single pass of the first two steps of the path-finding algorithm; it contains two segments: a chaotic trajectory at a higher parameter value that travels most of the way to the *other* fixed point and a short section of the non-chaotic spiral that surrounds the second fixed point. Note that this trajectory makes an initially-counterintuitive move directly away from its control objective. Not only has this manipulation allowed the trajectory to jump the basin boundary and converge to the opposite fixed point, but it has also bypassed much of the slowly-converging spiral around that point.

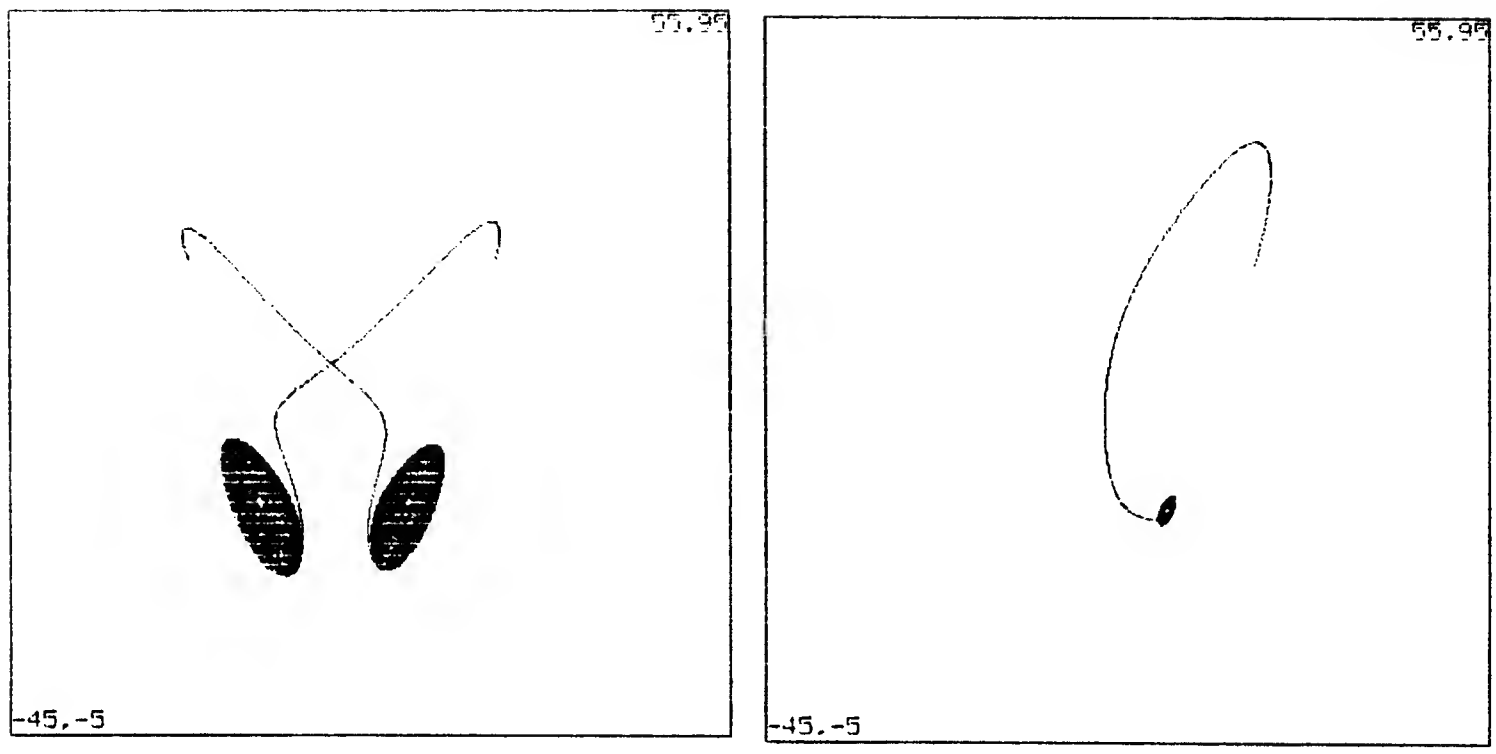

Figure 5: Segmented Path to a Fixed Point (from [2])

5 Summary

A chaotic system's behavior, and thus its state-space features, are strongly affected by parameter values. Moreover, the trajectories that make up those features separate exponentially over time. Small changes in parameters or state can thus have large and rapid effects; this leverage is a powerful tool for control algorithms. This paper demonstrates how fast and accurate computation can be used to synthesize paths through a chaotic system's state space that exploit this leverage to accomplish otherwise-impossible control tasks. Nonlinear dynamics provides the mathematical tools used to choose values, tolerances, heuristics and limits for the algorithms that select and piece together trajectory segments to create these paths. These algorithms find shorter and faster trajectories than traditional control methods, make unreachable control objectives reachable and improve convergence. They do this by constructing strange attractor bridges, making counterintuitive moves, and utilizing regions of sensitive dependence. However, this performance gain does have a cost: the complexity of the tasks that are executed by *Perfect Moment* and the accuracy demanded by the very sensitivity that gives the program its power make computation speed a vital issue.

Preliminary versions of these programs have been successfully tested on simulated models of several systems, but the ultimate goal of this project is the control of real physical devices. Versions of the driven single pendulum and the phase-locked loop discussed in this paper have been constructed and

are in the midst of being instrumented for control. A double pendulum is also under development, but its instrumentation is much more difficult. Much work will have to be devoted to system identification and observation issues — modeling error, sensor and actuator inaccuracy due to D/A and A/D conversion and time delay, etc, — before the gains demonstrated in these simulations can be fully realized in the corresponding physical systems.

Acknowledgements

The author is grateful for Hal Abelson's proofreading comments, Brian LaMacchia's Chaos Monitor board and support by an American Association of University Women (AAUW) dissertation fellowship.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-89-J-3202 and by the National Science Foundation under grant number MIP-9001651. The author is supported by an American Association of University Women dissertation fellowship.

This paper was presented in October 1991 at the First Experimental Chaos Conference in Washington D.C. Proceedings: World Scientific.

References

[1] H. Abelson, A. A. Berlin, J. Katzenelson, W. H. McAllister, G. J. Rozas, and G. J. Sussman. The Supercomputer Toolkit and its applications. Technical Report AIM 1249, M.I.T. Artificial Intelligence Lab, July 1990.

[2] E. Bradley. Control algorithms for chaotic systems. In *First European Conference on Algebraic Computation in Control*, Paris, March 1991. Lecture Notes in Control and Information Sciences, Springer-Verlag.

[3] D. D'Humieres, M. R. Beasley, B. Huberman, and A. Libchaber. Chaotic states and routes to chaos in the forced pendulum. *Physical Review A*, 26:3483, 1982.

[4] W. L. Ditto, S. N. Rauseo, and M. L. Spano. Experimental control of chaos. *Physical Review Letters*, 65:3211, 1990.

[5] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields.* Springer-Verlag, New York, 1983.

[6] C. S. Hsu. *Cell-to-Cell Mapping.* Springer-Verlag, New York, 1987.

[7] E. Ott, C. Grebogi, and J. A. Yorke. Controlling chaos. In *Chaos: Proceedings of a Soviet-American Conference.* American Institute of Physics, 1990.

[8] J. Rees and W. Clinger. The revised3 report on the algorithmic language Scheme. *ACM SIGPLAN Notices*, 21:37, 1986.

[9] K. M.-K. Yip. KAM: Automatic planning and interpretation of numerical experiments using geometrical methods. Technical Report AI-TR 1163, M.I.T. Artificial Intelligence Lab, August 1989.